Mastering Robotic Skills in Real Visual Worlds through Model-based Reinforcement Learning

Wei Zhu[1,*], Masashi Okada[2,*,†], and Tadahiro Taniguchi[2,3]

***Abstract*— In the present work, a model-based reinforcement learning (RL), namely DayDreaming, is applied to quickly and accurately learn the robotic skills in the physical worlds only with the image observation. The model-based RL methods with the image observation can outperform the pure model-free RL frameworks in video games and robotic simulators, it is however rare to validate their advantages in the real worlds. With the aim of broadly applying the model-based RL algorithm and validating its effectiveness in the real-world scenarios, we directly utilize the physical robotic arm to learn diverse skills via the real-time interaction. To significantly reduce the real-world interaction time, a state transition model is created to generate the long-horizon simulated samples for the RL training. Since the only observation is the high-dimensional image which is too complicated to be directly used as the input and output of the transition model, we leverage contrastive learning to encode the image observation into a low-dimensional latent feature vector, which can exempt the commonly-used decoder of image reconstruction that might possibly cause object vanishing. The real comparison implementations with a robotic arm learning three different skills, reaching, positioning and pushing, demonstrate that our method distinctly outperforms another model-based RL algorithm and a model-free RL approach with respect of the learning efficiency and operation accuracy, which indicates a feasible way of widely applying the RL framework in the image-based real worlds.**

I. INTRODUCTION

There are two crucial issues that limit the broad applications of reinforcement learning (RL) for the physical robots: (1) sim-to-real discrepancies which make it prohibitively challenging to directly deploy the simulated policy on the real robots with the same performance in simulation; (2) high-cost interaction that prohibits the direct training with the physical robots. A plenty of successful works about the robotic manipulation are limited in simulation to validate the novel RL algorithms since a considerable amount of samples can be quickly and easily generated [1]–[4], while very few successful applications [5] are deployed on the real robotic arms without simulation or any prior knowledge due to the time-consuming interaction. Instead, most of the practical implementations are transferred from simulation or based on expert experiences [6]–[10]. However, the simulated policy may fail in certain real-world scenarios because of the sim-to-real gap. In order to significantly improve the learning efficiency in either simulation or real environments, most of works partially or heavily rely on expert knowledge [8], [11]–[13]. Additionally, offline RL can acquire manipulation skills from the mixed datasets including both the expert and random experiences [14], [15]. However, gathering offline experiences is time-consuming and the policy learned from the offline datasets may be non-inclusive due to the problem of distribution drift. In contrast, the model-based RL method applied in this work directly masters various skills from the online learning without any prior knowledge and the skill policy can be generalized for the unexplored conditions in the training process.

As an effective way to improve the learning efficiency, the model-based RL framework is a feasible strategy to directly learn the robotic skills in the real worlds. The model can be regarded as a simulator, which can quickly imagine a great amount of long-horizon interaction without operating the real robots. A plenty of robotic applications have validated the high learning efficiency of the model-based RL algorithms with the low-dimensional observations [16], [17], it is however challenging and complicated to create and update a state transition model with the high-dimensional image observations being the input and output of the model because predicting the image might probably accumulate errors and consume considerable calculation resources.

When the robots need to manipulate objects such as the grasp and assembly tasks, and navigate in the dynamic environments like autonomous driving and the service in public places, the image observations are indispensable apart from the low-dimensional proprioceptive states, e.g. joint angles, gripper position, and vehicle velocity. For the model-free RL framework, the image observation are generally encoded and compressed by convolutional neural networks (CNNs) to extract an abstract low-dimensional feature vector at first [11], [18], which next concatenates other explicit information to form the RL states [19]–[21]. However, updating the complex CNNs and acquiring an optimal policy with broad generalization ability generally require a tremendous amount of exploration and interaction. For the model-based RL algorithms, one crucial issue is how to create a state transition model with the image observations. Recently, a recurrent state-space model (RSSM) successfully encodes the image observation into a latent state and predicts the future states in long horizon given the current state, action, and history information accumulated via the recurrent neural networks (RNNs) [22]. Since the samples used to update the RL policy is generated from the latent space and the state transition model, it is computationally cheap and can

[1]W. Zhu is with the Department of Robotics, Graduate School of Engineering, Tohoku University, Japan.

[2]M. Okada and T. Taniguchi are with Digital & AI Technology Center, Technology Division, Panasonic Corporation, Japan.

[3]T. Taniguchi is also with College of Information Science and Engineering, Ritsumeikan University, Japan.

[*]W. Zhu and M. Okada equally contribute to this work.

[†]Corresponding author: okada.masashi001@jp.panasonic.com

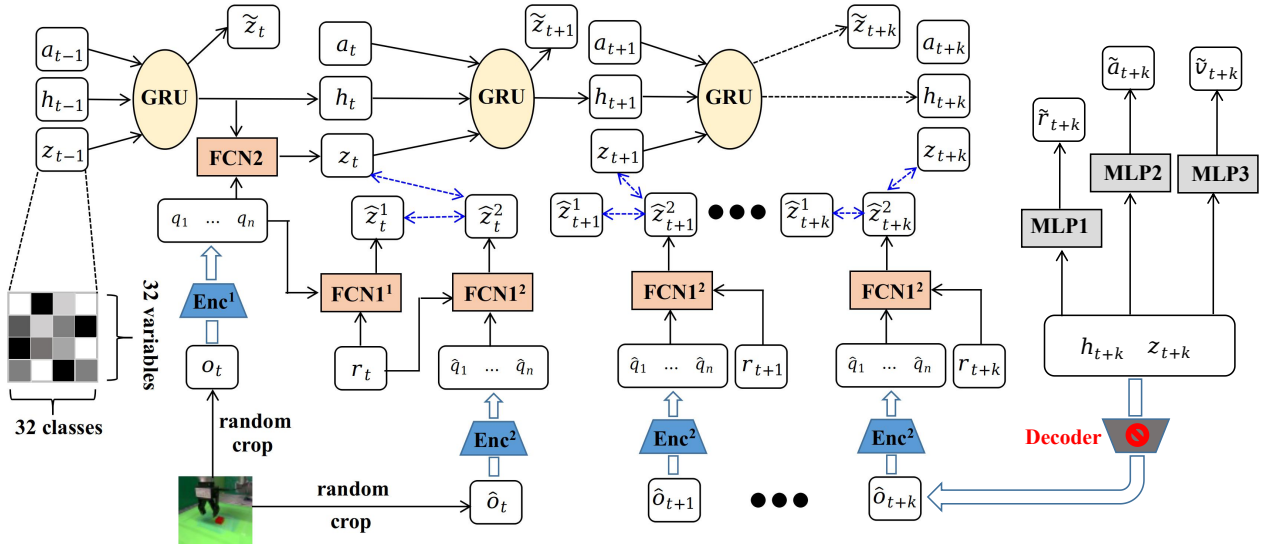


Fig. 1. The framework of model-based RL with the image observation. Enc^1 and Enc^2 represent two image encoders which have an identical CNN framework. FCN stands for an one-layer fully connected network while MLP is a multi-layer perceptron. GRU means a gate recurrent unit. The blue dotted two-way arrow points the pair of contrastive learning. In the real experiments, the image size is $64 \times 64 \times 3$, which is projected into a 32×32 discrete variable matrix used as the latent state in the transition model. The decoder of the variable matrix is exempted from our framework so as to relieve the problem of object vanishing.

improve the learning efficiency significantly [23], [24]. However, the image reconstruction required to update the RSSM networks may cause the problem of object vanishing [1], [25]. Therefore, as an alternative of encoding the image observation, contrastive learning [26] can be adopted to free the image construction [1], [6]. Nevertheless, the image-based RL algorithms are mainly limited in video games and robotic simulators due to the real-world difficulties of reward definition, object detection, environment resetting, and so forth.

Motivated by the above, we apply a model-based reinforcement learning (RL) method to quickly and accurately learn the robotic skills in the physical worlds only with the image observation. We name the method DayDreaming because it can acquire the skill by dreaming the virtual interaction via a state transition model simultaneously with the real interaction. We develop the DayDreaming with the core framework of the DreamingV2 algorithm [27] which demonstrates the outstanding performance in simulated environments. However, DreamingV2 assumes all the environment information is perfectly known such as the pose of the object and the image observation. For DayDreaming, we fine-tune certain hyper-parameters to make it more practical for the learning of the real robotic arm and define the real task scenarios based on the real and physical conditions. At the meanwhile, DayDreaming is different from Daydreamer [5] in that we free the process of the image reconstruction thus to diminish the problem of object vanishing. We are aimed at learning three tasks, reaching, positioning and pushing respectively. While reaching is a basic task used to validate a plenty of RL algorithms, positioning is one of the fundamental skills in factories, warehouses, retail stores, and so forth, which can be further combined with other related tasks, such as the insertion, fitting, pick and place [28]. Moreover, pushing is a rather more challenging task since the interaction between the robotic arm and the object is complex. Our primary contributions are summarized as follows:

- We directly train a physical robotic arm in the real worlds by leveraging a model-based RL framework that is efficient in sampling and precise for the robotic manipulation, which could be a feasible way for the practical applications of the RL algorithms.
- DayDreaming enables the robotic arm to learn diverse skills including reaching, positioning and pushing in the real worlds, which validates the generalization ability of the DreamingV2 algorithm not only in simulation but also in the real worlds.
- We compare DayDreaming with a state-of-the-art model-based RL method named DreamerV2 and an advanced model-free RL approach called DrQ-v2, with the ablation results experimentally indicating that DayDreaming could be a more suitable alternative for learning certain robotic skills.

II. APPROACH

The present approach, namely DayDreaming, is based on DreamerV2 [24], DayDreamer [5] and DreamingV2 [27]. Our framework leverages contrastive learning [26] to exempt the image reconstruction used in DreamerV2 and DayDreamer so as to diminish the influence of object vanishing caused by the image reconstruction [1]. Fig. 1 illustrates the top-level concept of our algorithm framework, with the details summarized in this section. We decouple the framework into two parts which are alternately updated: the world model learning and the RL policy optimization. The

world model is updated from the past experiences collected from the direct interaction in the real worlds. On the contrary, the samples used for the policy optimization are virtually generated via the world model, which can relieve too much physical operation. The updated policy is in turn deployed into the real worlds to further replenish the replay buffer of past experiences.

A. World Model Learning

Prediction model with the image observation. Directly predicting the next image observation based on the present image and action may bring about accumulated errors and require a tremendously amount of computer resources when the training batch size is large and the prediction horizon is long. Therefore, we leverage the idea of RSSM [22] to compress the high-dimensional image observation into a low-dimensional latent state, and have the multi-step prediction process implemented in the latent space. The prediction model is defined as follows:

$$h_t = f_\phi^h(h_{t-1}, a_{t-1}, z_{t-1}), \tag{1a}$$

$$\tilde{z}_t \sim p_\phi^z(\tilde{z}_t|h_t), \tag{1b}$$

$$z_t \sim q_\phi(z_t|h_t, o_t). \tag{1c}$$

h_t is a deterministic feature vector at time t which selects the possibly key information from the long-horizon historical experience and the last state as well as the last action. z stands for a latent state variable matrix with a low dimension. Given the last action a_{t-1}, the last latent state z_{t-1} and h_{t-1}, we can inference h_t and the current latent state $\tilde{z}_t$ from gated recurrent units (GRUs) [29] in the latent space. Additionally, a decoder q_ϕ is utilized to compress the image observation o_t into the latent state. Instead of directly representing the latent state z as a diagonal Gaussian [23], we utilize the categorical latent variables shown in Fig. 1 from the motivation that the categorical distributions can naturally capture multi-modal uncertainty of stochastic state transitions [27]. Take the pushing task implemented in this work for example, the contact between the end effector and the object is rich and complex, which makes the state transition model highly discontinuous. In this case, the discrete state representation with the categorical latent variables can experimentally behave better than the diagonal Gaussian representation [24], [27].

Distribution objective. One goal of the world model is to make the two stochastic distributions $\tilde{z}_t$ and z_t as close as possible, therefore, we define the first objective as a Kullback-Leibler divergence (KL) at time $t+i$ with $i = 0, 1, ..., k$:

$$\mathcal{J}_i^{KL} = KL[q_\phi(z_{t+i}|h_{t+i}, o_{t+i})||p_\phi(\tilde{z}_{t+i}|h_{t+i})]. \tag{2}$$

Contrastive objective. Instead of reconstructing the image from the latent state [22]–[24] to form the objective for unsupervised representation learning, we choose contrastive learning to complete the representation learning because it is empirically able to diminish the problem of object vanishing, which is important when the robotic arms operate small objects. In addition, we augment the data by the random crop technique to improve the unsupervised representation [21], which is also experimentally validated in the reaching task introduced in the experiment section.

We leverage the InfoNCE (noise contrastive estimator) [26] to define the contrastive objective at time $t+i$ with $i = 0, 1, ..., k$:

$$\begin{aligned}\mathcal{J}_i^{CO} = &\frac{\hat{z}_{t+i}^1 \otimes \hat{z}_{t+i}^2}{\sum_{j=0}^k \hat{z}_{t+j}^1 \otimes \hat{z}_{t+i}^2} \\ &+ \frac{\bar{z}_{t+i} \otimes \hat{z}_{t+i}^2}{\sum_{j=0}^k \bar{z}_{t+j} \otimes \hat{z}_{t+i}^2},\end{aligned} \tag{3}$$

with $\otimes$ being the Hadamard. To facilitate the contrastive unsupervised learning, we leverage the technique of momentum contrast which is widely used for unsupervised visual representation learning [30], [31]. Enc^1 and Enc^2, FCN1^1 and FCN1^2 shown in Fig. 1 are the two pairs used as the momentum contrast, which form the two items of the contrastive object (3).

Reward objective. Because the policy optimization is executed in the latent space, the reward can not be directly obtained from the real worlds while implementing the long-horizon prediction. Therefore, a neural network is required to predict the reward from the latent state, with the definition described as following:

$$\tilde{r}_t \sim p_\phi^r(\tilde{r}_t|z_t, h_t). \tag{4}$$

The reward objective at time $t+i$ can be simply defined as a log probability:

$$\mathcal{J}_i^R = \log p_\phi^r(r_{t+i}|z_{t+i}, h_{t+i}), \tag{5}$$

with r_{t+i} being the reward obtained from the real interaction at time $t+i$.

Variational evidence lower bound objective. The final overall objective used to update the world model can be defined as a variational evidence lower bound objective (ELBO):

$$\begin{aligned}\mathcal{J}^{ELBO} = &\mathbb{E}_{q_\phi(z_{t:t+k}|a_{t:t+k}, o_{t:t+k})} \sum_{i=0}^{k} (-\beta \mathcal{J}_i^{KL} \\ &+ \mathcal{J}_i^{CO} + \mathcal{J}_i^R),\end{aligned} \tag{6}$$

where β is used to scale the KL objective and the convergence proof of ELBO can be referred from [23], [26].

B. Policy Optimization

Since we have the inference models (1a) and (1b), we can quickly imagine a plenty of episodes in the latent space without operating the real robot. Assuming we have a critic network to map the latent state to the value function $\tilde{v}_t$ and an actor network to project the latent state to the action $\tilde{a}_t$:

$$\tilde{v}_t \sim q_\theta^v(\tilde{v}_t|\tilde{z}_t, h_t), \tag{7a}$$

$$\tilde{a}_t \sim q_\theta^a(\tilde{a}_t|\tilde{z}_t, h_t), \tag{7b}$$

with $\tilde{z}$ being the predicted latent state shown in Fig. 1. We can evaluate the value function via n-step RL because it

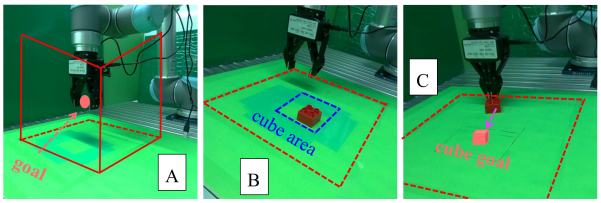


Fig. 2. Experimental scenes. The image observation is sampled from a fixed RGB camera whose view can cover the motion area of the end effector. A is the reaching task scenario where the initial position of the end effector is randomly set in a 3D space and the goal position is fixed for simplification. B represents the positioning task that the end effector randomly located in a safe motion area is required to move exactly above the cube object with the random position in a relatively small area. C illustrates the pushing task that the end effector should push the cube object to a goal position as close as possible.

has better unbiased estimation than one-step RL and is free of the tyranny of the single time step [32]. The discount accumulated return is defined as below:

$$V_N^i(s_\tau) \doteq \mathbb{E}_{(p_\phi^r, q_\theta)}\left[\sum_{n=\tau}^{h-1}(\gamma^{n-\tau}\tilde{r}_n) + \gamma^{h-\tau}\tilde{v}(s_h)\right], \quad (8)$$
$$V_\lambda(s_\tau) \doteq (1-\lambda)\sum_{n=1}^{H-1}(\lambda^{n-1}V_N^n(s_\tau)) + \lambda^{H-1}V_N^H(s_\tau),$$

with γ and λ being the discount factors and H standing for the prediction horizon length. Then we can intuitively define the value function objective (VO) as the sum of the log probabilities along with the prediction horizon:

$$\mathcal{J}^{VO} = \mathbb{E}_{(f_\phi^h, p_\phi^z, p_\phi^r, q_\theta)}\left[\sum_{\tau=t}^{t+H-1}\log q_\theta^v(V_\lambda(s_\tau)|s_\tau)\right]. \quad (9)$$

At the meanwhile, the action needs to be optimized to maximize the value function, therefore, we represent the action objective (AO) as:

$$\mathcal{J}^{AO} = \mathbb{E}_{(f_\phi^h, p_\phi^z, p_\phi^r, q_\theta)} \left[\sum_{\tau=t}^{t+H-1}\log q_\theta^a(a_\tau|s_\tau)\cdot(\tilde{v}_\tau - V_\lambda(s_\tau))\right]. \quad (10)$$

C. Implementation

We set the sampling time length as $k = 32$ while the prediction horizon length in the latent space is given as $H = 15$. The batch size is 32. The corresponding factors β, γ, and λ are 1.0, 0.99, and 0.95 respectively. We use the Adam optimizer to update the world model and the RL model with all learning rates being the same as 0.0001.

III. EXPERIMENTS

We performed three tasks, namely reaching, positioning and pushing respectively, with the experimental scenes illustrated in Fig. 2. Moreover, we executed three baselines to highlight the advantages of our model-based RL framework, namely DayDreaming. One is a purely model-free RL framework similar to the deep deterministic policy gradient (DDPG) algorithm used in [19]. However, we found that the objective loss tended to be infinite even though we carefully tuned the hyper-parameters. Therefore, we omitted this baseline in the quantitative analysis. Another one is a data-augmented model-free RL approach derived from [21], namely DrQ-v2. The last one is a state-of-the-art model-based RL originated from [5], [24], called DreamerV2. All the tasks share the same size of the image observation $64 \times 64 \times 3$ but have slightly different reward definitions and motion areas, with the details shown as follows.

A. Reaching

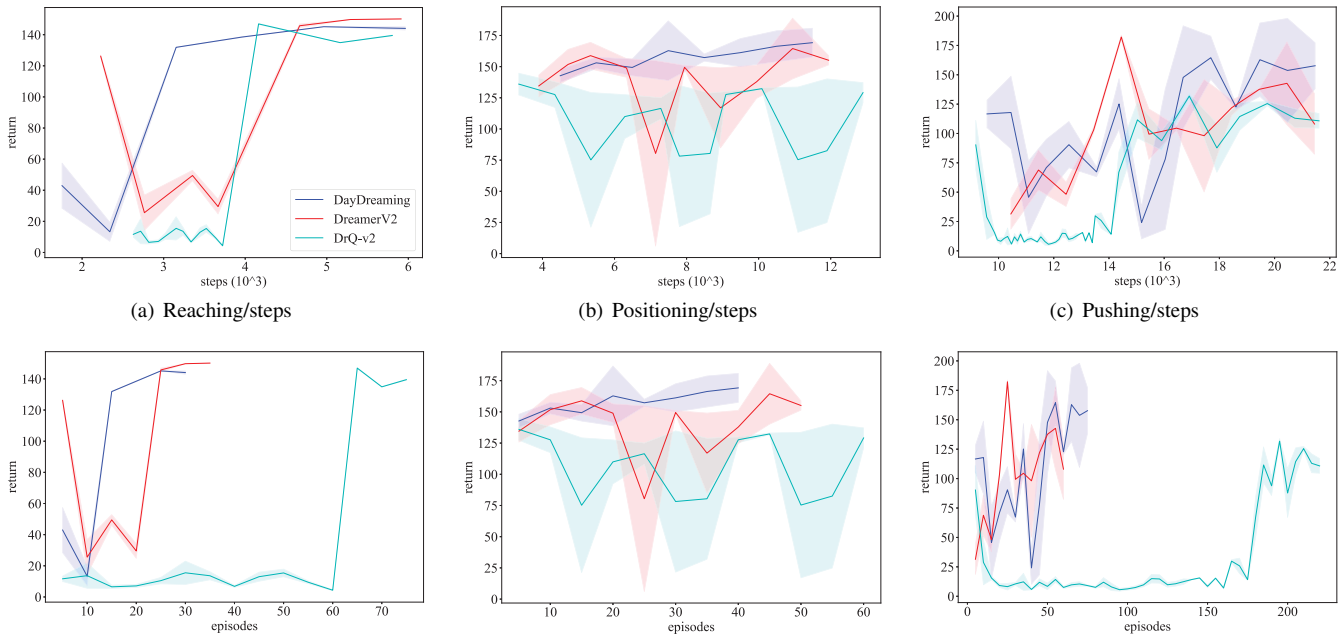
For the basic task of reaching, we fix the goal position P_g for simplification in a 3D space $0.26 \times 0.26 \times 0.14$m. When the end effector moves out of the area, a penalty of -0.1 is immediately given and a new episode starts. After the time step reaches 200, the episode is also resumed. At the beginning of each episode, the position of the end effector P_e is randomly set in the specified area. If the end effector is within the area during the interaction, the reward is defined as follows:

$$r = (1 - d_g/d_m)/1.25, \text{ if } d_g > 0.007 \text{ else } 1, \quad (11)$$

with $d_g = ||P_g - P_e||_2$ being the distance from the end effector to the fixed goal and d_m being the diagonal length of the area. We specify a minimum threshold of 7mm to give a high reward as 1 around the goal due to the low resolution of the image observation, thus to enable less stochastic motion when the end effector reaches the goal. The action space is composed by 3 continuous velocities in the x/y/z directions of the world coordinate, with the range [-0.05, 0.05] m/s and the control frequency 10Hz.

We firstly initialized the experience pool using 30 episodes and pre-trained the model 2000 times. Instead of individually maintaining a thread to update the model, we trained the model 200 times at the end of each episode to reduce the delay caused by the thread communication. Moreover, during the model updating, we could have enough time to reset the episode. Because the episode was terminated when the end effector moved out of the specified area, the episode length varied accordingly. The training process is shown in Fig. 3(a) and 3(d). After the training, we evaluated 20 times, with the quantitative results shown in TABLE. I.

As shown in Fig. 3(a), our approach could quickly learn the reaching skill within 3000 time steps which were equal to 5 minutes in the real interaction. Although the DreamerV2 method obtained the slightly higher average return of the final 20 evaluations, it failed 8 times whose returns were omitted to calculate the average return. Because our method leveraged the data augmentation technique to process the image margin and avoided the image decoder via contrastive learning, the end effector was able to stably move within the specified area. Additionally, the model-free RL framework DrQ-v2 required rather more episodes but produced the lowest average return, which validated that our model-based RL method and the baseline DreamerV2 as another model-based RL approach were able to improve the learning efficiency of RL and the operation accuracy for the reaching task.



(a) Reaching/steps (b) Positioning/steps (c) Pushing/steps

(d) Reaching/episodes (e) Positioning/episodes (f) Pushing/episodes

Fig. 3. Training processes of all the three tasks. Three evaluations are implemented every five training episodes and these figures depict the evaluation returns along with the time steps and the interaction episodes. The return is the sum of all rewards corresponding to the time steps in one episode. The upper figures illustrate the evaluation return with the time steps while the bottom ones show the assessment with the episodes. (a)(d), (b)(e) and (c)(f) represent the training processes of the reaching, positioning and pushing tasks respectively. The time steps are proportional to the real interaction time with the scale factor about 0.1 second. More training episodes bring about more training steps because 200 iterations are executed to update the neural networks after each episode. In addition, more frequent resetting operations are required with the episode number increasing, which significantly adds the extra time of the learning process. Certain episodes have less than 200 steps because the end effector moves out of the area and the episodes terminate ahead of the maximum time steps.

TABLE I

FINAL EVALUATION OF THE REACHING TASK.

	DayDreaming	DreamerV2	DrQ-v2
out-of-area times	**2**	8	6
average return	148.5	**152.7**	141.9
time steps	5900	5900	**5800**
episodes	**30**	35	75

B. *Positioning*

We progressively increased the task difficulty. For the positioning task shown in Fig. 2, the end effector is required to move above a small cube object for the lift task. Please note that we do not directly acquire the lift skill with a single framework so as to reduce the task complexity. Additionally, the initial position of the end effector is randomly set on a 0.26×0.26m area while the cube object is randomly located on a 0.05×0.05m square. To further simplify the problem, we assume that the end effector moves on a 2D plane rather than in a 3D space and the rotation angle of the cube object is approximately a constant. Similar to the reaching task, when the end effector moves out of the area, it is punished by -0.1. Besides, the reward will be 1 if the end effector is above the cube object within a threshold d_t. Let P_g be the position of the cube object. Before each episode, we can roughly calibrate P_g by manually locating the end effector above the cube object. Therefore, the reward definition within the motion area is same as the reaching task's (11). Although calibration is inaccurate, the final learned policy can still achieve the positioning task with high success rate. Please note that we do not use the camera to calculate the precise position of the cube because the camera calibration takes time every time the camera pose changes. Additionally, the view of the cube may be blocked by the end effect during the exploration process.

After initializing the experience pool with 30 episodes generated by the random policy, we pre-trained the model 2000 times to obtain a rough policy. The training process and final evaluation are shown in Fig. 3(b) and 3(e), and TABLE. II respectively. In Fig. 3(b), both the model-based approaches could quickly obtain a feasible policy with about 20 minute real interaction and less than 40 episodes. However, our method could generate higher return at the beginning and keep a stable training process while the DreamerV2 approach produced a fluctuated learning curve shown in Fig. 3(b). The same problem of vibration obviously occurred in DrQ-v2. Additionally, as the final 20 evaluations show, the DreamerV2 method was unable to well deal with the image margin since it failed 4 times. The task failure includes two cases: the end effector either moves out of the area or is always far from the cube object. Although our approach failed one time, it could accurately keep the end effector within the

motion area and obtain a higher average return shown in TABLE. II. On the contrary, the model-free baseline DrQ-v2 was significantly less precise since it failed all evaluations and produced the lowest average return.

TABLE II
FINAL EVALUATION OF THE POSITIONING TASK.

	DayDreaming	DreamerV2	DrQ-v2
out-of-area times	**0**	4	0
failures	**1**	4	20
average return	**165.9**	158.5	134.3
time steps	**11500**	11900	12900
episodes	**40**	50	60

C. Pushing

We finally implemented a more challenging task, pushing a cube object to a goal area on a 2D plane. The initial position of the end effector is randomly set on the back one third of the motion area while the cube object is put in front of the end effector so that the camera can view the cube without being blocked by the end effector. We do not roughly calibrate the position of the cube with the robotic arm as the positioning task because the cube has complex motion when it is pushed. Instead, we first detect the cube object and then get its centroid which corresponds to the pixel coordinate of the image observation. Please note that we do not use the absolute position relative to the frame of the robotic arm simply for simplification, otherwise, we need to calibrate the camera and design an absolute positioning method which is out of the scope of our work. The reward definition is still similar to that of the reaching task (11). Because the cube centroid detected by the camera is influenced by the perspective principle, the whole experiment may be inaccurate. However, the training process and the final evaluations can still demonstrate the advantages of our method.

As the results shown in Fig. 3(c) and 3(f), TABLE. III, DayDreaming and DreamerV2 significantly outperformed the model-free framework in respect of learning efficiency. DrQ-v2 required far more episodes to obtain a feasible policy, which brought about very frequent resetting procedure. Moreover, DayDreaming was able to achieve rather more precise performance with respect to the average return. We define the failure if the episode return is less than 90 or the motion is out of the area. Although our method failed 3 times, we found that 2 failures were caused by the view blocking, that is the reward was zero if the cube could not be detected due to the blocking. Even with long-time interaction (about 36 minutes), DreamerV2 failed 7 times due to the out-of-area motion and small episode returns. We think that the problem was possibly caused by the decoder used in DreamerV2 which might vanish the cube object.

TABLE III
FINAL EVALUATION OF THE PUSHING TASK.

	DayDreaming	DreamerV2	DrQ-v2
out-of-area times	**0**	1	0
failures	3	7	**1**
average return	**152.8**	129.9	110.6
time steps	21500	**21400**	21600
episodes	95	**80**	235

D. Experiment Summary

We performed three tasks, reaching, positioning and pushing respectively and compared our method with another two state-of-the-art RL algorithms only with the image observation. The interaction time used to learn these skills was 10, 20 and 36 minutes respectively. Moreover, the episodes were 30, 40 and 95, which were equal to the episodes of another model-based RL approach and rather less than a model-free RL framework. With the same interaction time and the limited time steps, our method could achieve higher success rate and guarantee more precise performance even with the low-resolution image, which could be an effective learning framework in the real worlds only with the image observation.

IV. CONCLUSIONS

In this work, we applied a model-based RL framework to learn robotic skills directly in the real worlds, thus to avoid the sim-to-real gap when deploying the RL policies on the physical robots. Aimed at reducing the real interaction, we created a state transition model as the simulator which could quickly generate long-horizon samples for the RL training without operating the real robot. To encode and compress the high-dimensional image observation without the image reconstruction, we utilized the technique of data augmentation and contrastive learning, which experimentally validated that our method could deal with the margin conditions and diminish the problem of object vanishing.

Although we have achieved three robotic skills, reaching, positioning and pushing respectively, it is still challenging to apply our algorithm framework in more complex scenarios. On the one hand, the motion area and the initial states in our present work are limited in small spaces for simplification. When we relieve these limitations, the interaction time increases considerably. On the other hand, the size of the work space is also restricted by the low-resolution image. If we increase the resolution to cover a larger area, we need longer training time and more computation resources. Additionally, for the manipulation task, it is difficult to define an accurate reward function which is crucial for RL algorithms but cumbersome to obtain in the real worlds. In our future works, we will focus on solving these limitations to enable more practical robotic applications with the model-based RL framework.

REFERENCES

[1] M. Okada and T. Taniguchi, "Dreaming: Model-based Reinforcement Learning by Latent Imagination without Reconstruction," in *ICRA*, 2021.

[2] A. Byravan, J. T. Springenberg, A. Abdolmaleki, R. Hafner, M. Neunert, T. Lampe, N. Siegel, N. Heess, and M. Riedmiller, "Imagined Value Gradients: Model-based Policy Optimization with Tranferable Latent Dynamics Models," in *Conference on Robot Learning (CoRL)*, 2020.

[3] N. Hansen and X. Wang, "Generalization in Reinforcement Learning by Soft Data Augmentation," in *ICRA*, 2021.

[4] S. James and A. J. Davison, "Q-Attention: Enabling Efficient Learning for Vision-Based Robotic Manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1612-1619, 2022.

[5] P. Wu, A. Escontrela, D. Hafner, K. Goldberg, and P. Abbeel, "DayDreamer: World Models for Physical Robot Learning," *arXiv preprint arXiv:2206.14176*, 2022.

[6] W. Yan, A. Vangipuram, P. Abbeel, and L. Pinto, "Learning Predictive Representations for Deformable Objects using Contrastive Estimation," *arXiv preprint arXiv:2003.05436*, 2020.

[7] S. James, A. J. Davison, and E. Johns, "Transferring End-to-end Visuomotor Control from Simulation to Real World for a Multi-stage Task." in *Conference on Robot Learning (CoRL)* 2017.

[8] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramar, R. Hadsell, N. Freitas, and N. Heess, "Reinforcement and Imitation Learning for Diverse Visuomotor Skills," *arXiv preprint arXiv:1802.09564*, 2018.

[9] D. Son, M. Kim, J. Sim and W. Shin, "Reinforcement Learning for Vision-based Object Manipulation with Non-parametric Policy and Action Primitives," in *IROS*, 2021.

[10] G. Schoettler, A. Nair, J. A. Ojea, S. Levine, and E. Solowjow, "Meta-Reinforcement Learning for Robotic Industrial Insertion Tasks," in *IROS*, 2020.

[11] G. Schoettler, A. Nair, J. Luo, S. Bahl, J. A. Ojea, E. Solowjow, and S. Levine, "Deep Reinforcement Learning for Industrial Insertion Tasks with Visual Inputs and Natural Rewards," in *IROS*, 2020.

[12] T. Z. Zhao, J. Luo, O. Sushkov, R. Pevceviciute, N. Heess, J. Scholz, S. Schaal, and S. Levine, "Offline Meta-Reinforcement Learning for Industrial Insertion," in *ICRA*, 2022.

[13] A. Zhan, R. Zhao, L. Pinto, P. Abbeel, M. Laskin, "A Framework for Efficient Robotic Manipulation," *arXiv preprint arXiv:2012.07975*, 2020.

[14] R. Rafailov, T. Yu, A. Rajeswaran, and C. Finn, "Offline Reinforcement Learning from Images with Latent Space Models," in *Learning for Dynamics and Control (L4DC)*, 2021.

[15] T. Yu, A. Kumar, R. Rafailov, A. Rajeswara, S. Levine, and C. Finn, "Combo: Conservative Offline Model-based Policy Optimization," in *Advances in Neural Information Processing Systems (NIPS)*, 2021.

[16] N. O. Lambert, D. S. Drew, J. Yaconelli, S. Levine, R. Calandra and K. S. J. Pister, "Low-Level Control of a Quadrotor with Deep Model-Based Reinforcement Learning," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4224-4230, 2019.

[17] T. G. Thuruthel, E. Falotico, F. Renda and C. Laschi, "Model-based Reinforcement Learning for Closed-Loop Dynamic Control of Soft Robotic Manipulators," *IEEE Transactions on Robotics*, vol. 35, no. 1, pp. 124-134, 2019.

[18] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez and T. Funkhouser, "Learning Synergies Between Pushing and Grasping with Self-Supervised Deep Reinforcement Learning," in *IROS*, 2018.

[19] J. Jin, N. M. Nguyen, N. Sakib, D. Graves, H. Yao and M. Jagersand, "Mapless Navigation among Dynamics with Social-safety-awareness: A Reinforcement Learning Approach from 2D Laser Scans," in *ICRA*, 2020.

[20] I. Kostrikov, D. Yarats, and R. Fergus, "Image Augmentation is All You Need: Regularizing Deep Reinforcement Learning from Pixels," in *International Conference on Learning Representations (ICLR)*, 2020.

[21] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto, "Mastering Visual Continuous Control: Improved Data-augmented Reinforcement Learning," *arXiv preprint arXiv:2107.09645*, 2021.

[22] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning Latent Dynamics for Planning from Pixels," *arXiv preprint arXiv:1811.04551*, 2018.

[23] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to Control: Learning Behaviors by Latent Imagination," *arXiv preprint arXiv:1912.01603*, 2019.

[24] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Mastering Atari with Discrete World Models," *arXiv preprint arXiv:2010.02193*, 2020.

[25] F. Deng, I, Jang, and S. Ahn, "Dreamerpro: Reconstruction-free Model-based Reinforcement Learning with Prototypical Representations," in *International Conference on Machine Learning (ICML)*, 2022.

[26] A. Oord, Y. Li, and O. Vinyals, "Representation Learning with Contrastive Predictive Coding," *arXiv preprint arXiv:1807.03748*, 2018.

[27] M. Okada and T. Taniguchi, "DreamingV2: Reinforcement Learning with Discrete World Models without Reconstruction," in *IROS*, 2022.

[28] R. Okumura, N. Nishio, and T. Taniguchi, "Tactile-Sensitive NewtonianVAE for High-Accuracy Industrial Connector-Socket Insertion," in *IROS*, 2022.

[29] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[30] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum Contrast for Unsupervised Visual Representation Learning," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[31] A. Srinivas, M. Laskin, and P. Abbeel, "Curl: Contrastive Unsupervised Representations for Reinforcement Learning," *arXiv preprint arXiv:2004.04136*, 2020.

[32] R. S. Sutton and G. B. Andrew, "Reinforcement Learning: An Introduction," *MIT press*, 2018.